Görev-6

ASP.NET Core WebAPI Öğrenci Bilgi Sistemi

Amaç:

Bu görevde, .NET Core Web API kullanarak bir öğrenci bilgi sistemi tasarlamak ve bu sistemi geliştirmek amaçlanmaktadır. Bu sistem, öğrenci bilgilerini almak, kaydetmek ve doğrulamak için çeşitli teknikleri içerecektir. Sayfaların görsel tasarımı, model ve form içerikleri sizin tarafınızdan detaylandırılabilir.

Yapılacaklar:

- **Proje Oluşturma:**
  Visual Studio kullanarak bir .NET Core WebAPI projesi oluşturun.
- **Model Oluşturma:**
  Bir Student sınıfı oluşturun. Bu sınıf, öğrenci bilgilerini temsil etmelidir (örneğin, öğrenci adı, numarası, sınıfı, vb.).
- **Controller Oluşturma:**
  Bir StudentsController oluşturun. Bu controller, öğrenci bilgilerini işleyecek CRUD (Create, Read, Update, Delete) operasyonlarını içermelidir.
- **Model Binding:**
  Bir öğrenci eklemek için POST metodu oluşturun ve bu metodun model binding ile çalışmasını sağlayın. Öğrenci bilgilerini alın, doğrulayın ve static bir listeye ekleyin. CRUD işlemleri için diğer action'ları da uygun şekilde oluşturun.
- **Server Side Validasyon:**
  Server side validasyonları ekleyin;
  - Öğrenci adı ve soyadı alanları boş olmamalıdır.
  - Öğrenci numarası benzersiz olmalıdır (static listeye daha önce eklenmemiş olmalıdır).
- **HTML Form ve Client Side Validasyon:**
  CRUD işlemlerinden Create ve Update için form elementi kullanılabilir. Fakat Read ve Delete işlemlerini fetch ile yapmanız gerekecektir.
  WebAPI projesi ve HTML formlarını barındıran sayfalar farklı klasörlerde hazırlanmalıdır.
  JavaScript kullanarak form üzerinde client side validasyonları sağlayın. Örneğin, boş alanlar, benzersizlik kontrolü gibi.

WebAPI ve HTML Formları Arasındaki İlişki:

HTML formu ile Web API arasındaki ilişkiyi kurmak için şu adımları takip edebilirsiniz:

- **HTML Formu:**
  Web sayfasında Create ve Update işlemleri için formlar oluşturulmalı ve bu form, kullanıcının öğrenci bilgilerini girmesine olanak tanımalıdır. Formun action ve method özellikleri, Web API'ye veri gönderimini yönlendirmelidir. Bunun için aşağıdaki örnekten faydalanabilirsiniz;

```
<form id="studentForm" action="https://localhost:5001/api/students" method="post">
    <!-- Form alanları buraya eklenecek -->
    <button type="submit">Öğrenci Ekle</button>
</form>
```

Form elementi ile sadece **get** ve **post** metodları kullanılabildiğinden, CRUD işlemleri için aşağıdaki yöntemleri izleyebilirsiniz;

- **Create(POST) =>** İlgili form doldurulup, form elementinin **post** metoduyla veriler gönderilebilir.
- **Read(GET) =>** Javascript fetch ile istekte bulunup, gelen request ile sayfadaki içerik dinamik olarak doldurulabilir. (Örn: bir tablo içerisinde)
- **Update(PUT) =>** İlgili form doldurulup, fetch'in **put** metodu ile gönderilebilir. (fetch ile **put** metodunun nasıl kullanıldığını araştırınız.)
- **Delete(DELETE) =>** fetch ile bir DELETE isteği yapılabilir. (fetch ile **delete** metodunun nasıl kullanıldığını araştırınız.)

- **JavaScript ile Client Side Validasyon:**
  JavaScript kullanarak form alanlarındaki değerleri kontrol edin ve gerekli client side validasyonları uygulayın.

```
document.getElementById("studentForm").addEventListener("submit", function (event) {
    // JavaScript ile client side validasyonları uygula
    if (!validateForm()) {
        event.preventDefault(); // Formun sunucuya gönderilmesini engelle ve uyarı göster
    }
});

function validateForm() {
    // Gerekli validasyonları uygula ve hataları kullanıcıya göster
    // Örneğin, boş alan kontrolü ve benzersizlik kontrolü
}
```

Teslim: WebAPI Uygulamasını ve formları barındıran sayfaları içeren klasörleri ayrı ayrı zip'leyin ve Görev-6 paylaşımına ek olarak ekleyin. WebAPI projesi ve HTML formlarını barındıran sayfalar farklı klasörlerde olmalı, toplamda iki adet zip dosyası teslim etmeniz gerekmektedir.